XP-002140936

2000 P 15376 EP

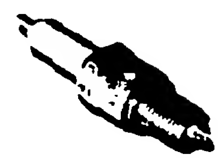
P.D. 1-07-1999
p. 71-80 10

AUTOCONFIGURATION

SERVICE LOCATION PROTOCOL: Automatic Discovery of IP Network Services

ERIK GUTTMAN, *Sun Microsystems*

As computers become more portable and networks larger and more pervasive, the need to automate the location and client configuration for network services also increases. The Service Location Protocol is an IETF standard that provides a scalable framework for automatic resource discovery on IP networks.



The complexity of configuring every element in the network—clients, servers, peers, and infrastructure—is a key problem facing network technology's advance. As long as configuration remains difficult, network administration will be expensive, tedious, and troublesome, and users will be unable to take advantage of the full range of capabilities networked systems could provide. The Service Location Protocol[1] is an Internet Engineering Task Force standard for enabling network-based applications to automatically discover the location—including address or domain name and other configuration information—of a required service. Clients can connect to and make use of services using SLP. Currently, without SLP, service locations must be manually configured or entered into a configuration file. SLP provides for fully decentralized operation and scales from small, unadministered networks to large enterprise networks with policies dictating who can discover which resources.

This article describes SLP's operation and how it adapts to conditions where infrastructure is not available, where administration is minimal, or where network administrators simply wish to reduce workload.

BACKGROUND

The Service Location Protocol (SVRLOC) working group has been active in the IETF for several years. In 1997, the group published SLP Version 1 as a Proposed Standard RFC.[1] In June 1999, the Internet Engineering Steering Group announced that Version 2 and its related documents were promoted to Proposed Standard RFCs as well.[2] SLPv2, which updates and replaces SLPv1, is the subject of this article. It removes several of the originally imposed requirements, provides protocol extensibility (new options can be added without modifying the base protocol), adheres to new IESG protocol recommendations, improves security, and eliminates a number of inconsistencies in the SLPv1 specification.

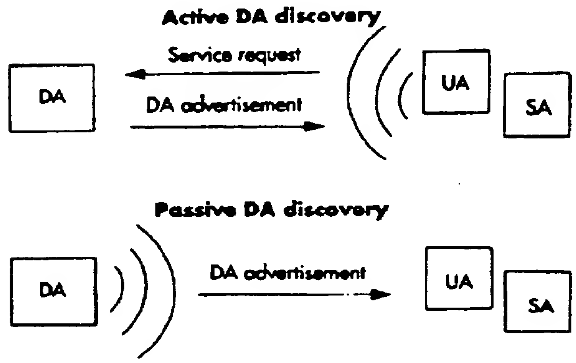


Figure 1. Methods of DA discovery. In active discovery, User Agents and Service Agents multicast requests to locate Directory Agents on the network, whereas, in passive discovery, UAs and SAs learn of DAs via periodic multicast advertisements.

Backward compatibility with SLPv1 depends on whether the Directory Agent (described in the next section) supports both versions. For example, Sun has successfully implemented a backwardly compatible SLP DA. Otherwise, backward compatibility requires a Service Agent (also described below) to implement both versions of the protocol.

Problems with Earlier Protocols

Prior to SLP, service discovery protocols allowed users to discover services only by type. For instance, both Apple and Microsoft offered networking protocols that could discover instances of printers and file servers, and users had to then select from the list to meet their needs. From the beginning, the SVRLOC working group sought a solution that would allow network software to discover services according to their characteristics as well as type. Thus, clients would be able to explicitly discover services that met their requirements, and software could automatically obtain the service location without bothering users.

On the other hand, since services are advertised along with their characteristics, SLP also enables rich user interaction. SLP enables browser operations since the protocol includes a set of directory-like functions. Thus, clients using SLP can browse all the available types of service. These clients may also request the attributes of a class of service, which aids in formulating interactive requests. Finally, SLP makes it possible to look up the attributes of a particular service once it has been discovered.

Another problem with the proprietary protocols was their notorious lack of scalability. The SRVLOC working group sought to correct this problem by minimizing the impact of service discovery on the network. SLP uses multicast and Dynamic Host Configuration Protocol[3] to initialize its scalable service discovery framework without the need for configuring individual SLP agents. SLP can operate in networks ranging from a single LAN to a network under a common administration, also known as an *enterprise network*. These networks can be quite large (potentially tens of thousands of networked devices). Neither multicast discovery nor DHCP scales to the Internet, since these protocols must be configured and administered. Moreover, the Internet lacks a common centralized administration. To the extent that SLP relies on either multicast discovery or DHCP for its own configuration, SLP does not scale to the Internet.

Current SLP Implementations

Sun Microsystems, Novell, IBM, Apple, Axis Communications, Lexmark, Madison River Technologies, and Hewlett-Packard have adopted SLPv1, and, increasingly, SLPv2 for products. There are also two reference implementations of SLPv2 available from http://www.srvloc.org/.

PROTOCOL OVERVIEW

SLP establishes a framework for resource discovery that includes three "agents" that operate on behalf of the network-based software:

- User Agents (UA) perform service discovery on behalf of client software.
- Service Agents (SA) advertise the location and attributes on behalf of services.
- Directory Agents (DA) aggregate service information into what is initially a stateless repository.

Figure 1 illustrates the two different methods for DA discovery: active and passive. In active discovery, UAs and SAs multicast SLP requests to the network. In passive discovery, DAs multicast advertisements for their services and continue to do this periodically in case any UAs or SAs have failed to receive the initial advertisement.

UAs and SAs can also learn the locations of DAs by using the DHCP options for Service Location, SLP DA Option (78).[4] DHCP servers, configured by network administrators, can use DHCP Option 78 to distribute the addresses of DAs to hosts that request them. SLP agents configured in this man-

ner do not require the use of multicast discovery, since this is only used to discover DAs and to discover services in the absence of DAs.

Operational Modes

SLP has two modes of operation:

- When a DA is present, it collects all service information advertised by SAs, and UAs unicast their requests to the DA.
- In the absence of a DA, UAs repeatedly multicast the same request they would have unicast to a DA. SAs listen for these multicast requests and unicast responses to the UA if it has advertised the requested service.

When a DA is present, UAs receive faster responses, SLP uses less network bandwidth, and fewer (or zero) multicast messages are issued.

Aside from unsolicited announcements sent by DAs, all messages in SLP are requests that elicit responses. By default, SLP agents send all messages in UDP datagrams; TCP is used only to send messages that don't fit in a single datagram.

Service Advertisements

Services are advertised using a Service URL, which contains the service's location: the IP address, port number and, depending on the service type, path. Client applications that obtain this URL have all the information they need to connect to the advertised service. The actual protocol the client uses to communicate with the service is independent of SLP.

Service Templates[5]—documents registered with the Internet Assigned Numbers Authority (IANA)—define the attributes associated with service advertisements. Templates specify the attributes, and their default values and interpretation, for a particular service type. SAs advertise services according to attribute definitions in the Service Templates, and UAs issue requests using these same definitions. This ensures interoperablility between vendors because every client will request services using the same vocabulary, and every service will advertise itself using well-known attributes.

OPERATIONS AND SCENARIOS

SLP operates in several different scenarios.

Initialization

At startup, UAs and SAs first determine whether there are any DAs on the network. DA addresses

In SLP, User Agents and Service Agents can use the multicast convergence algorithm to discover Directory Agents. UAs can also use it to issue requests when no DA is present. This algorithm allows SLP agents to receive replies from more responders than they could with standard multicast. Ordinarily in multicast, if there are many responders, a requester is likely to be inundated by the implosion of responses.

The SLP agent attempting to discover services (or DAs) issues an SLP Service Request message using multicast. This may result in one or more unicast response messages. After a wait period, the request is reissued with an appended "previous responders list," including the address of each SLP agent that has already responded. When SAs or DAs receive requests, they first examine the previous responder list. If they discover themselves on the list, they do not respond to the request.

The previous responder list can contain about 60–100 entries before it, combined with the request, becomes too large to fit in the request datagram. Reliable multicast is a notoriously difficult problem, but SLP's multicast convergence algorithm provides a "semi-reliable" multicast transaction.

Figure A illustrates the algorithm's operation as used in DA discovery. When the request is first sent, DAs 1, 2, and 3 reply, but the reply from DA 3 is lost. When the request is retransmitted a second time, DAs 1 and 2 do not respond, but since DA 3 is not yet on the list, it replies again. On the third retransmission no DAs respond since they all finally appear on the previous responders list.

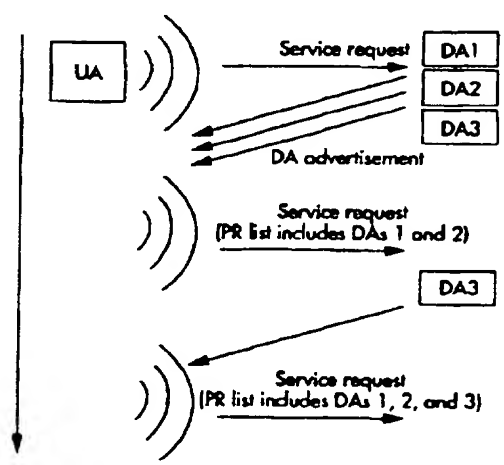


Figure A. The multicast convergence algorithm in DA discovery.

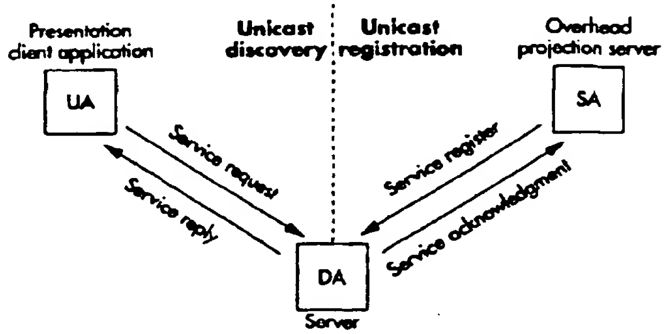


Figure 2. Normal operation of SLP. The SA registers the overhead projection server's location with the DA, and in response to the UA's Service Request message, it obtains this location from the DA via a Service Reply message.

can be configured statically (either manually configured, read from a configuration file, or hard-coded). DA locations can also be obtained dynamically using DHCP as discussed above. In these cases, there is no need to perform DA discovery. In all other cases, UAs and SAs use the SLP multicast convergence algorithm to discover DAs (see the sidebar, "SLP Multicast Convergence Algorithm"). They multicast Service Request messages to obtain DA advertisement messages, which include the Service URL[5] as well as the DA's scope, attributes, and digital signature. From this information UAs and SAs can locate the correct DA for message exchange.

Standard Case

Figure 2 depicts SLP's normal operation. In this example, a client program seeks an overhead projection server to display a presentation to an assembled audience. The SA registers the service's location with the DA, and the UA obtains this location from the DA in a Service Reply message.

Note that the SLP agents depicted may or may not reside on separate networked computers, but only one DA or SA can be on any given machine, due to the rules of the multicast convergence algorithm.

The networked service process advertises itself by registering with a DA, using an internal Service Location Protocol API.[6] An SA on the same computer as the network service registers service information by sending a Service Registration message to the DA and awaiting a Service Acknowledgment in reply.

Service registrations have lifetimes no greater than 18 hours, so the SA must reregister the service periodically, or the lifetime expires. In the event that the service terminates, the SA can optionally send a Service Deregister message to the DA; but even in the worst case, when the service fails, the registration will age out. This ensures that stale information will not persist with the DA.

Client software can use the standard Service Location Protocol API to find the particular service it requires. In this case, a UA sends the DA a Service Request that includes a search filter that is syntactically identical to the request format used by version 3 of the Lightweight Directory Access Protocol.[7] SLP thereby provides a directory-like lookup of all services that match the client's requirements. The DA returns a Service Reply message containing Service URLs and enough information for the client to contact each service that matches the request.

Larger Network Environments

A DA may serve thousands of clients and servers, so SLP gives administrators ways to improve overall performance and scalability of an SLP deployment as DAs become more loaded.

More DAs. First of all, administrators can simply activate more DAs to enhance SLP performance. Because SAs register with each DA they detect, all DAs will eventually contain the same service information, provided that all SAs can find them all. (Of course, sometimes this is neither possible nor desirable.) Adding more DAs creates roughly duplicate repositories of service information without requiring any formal database synchronization between them. Moreover, since UAs can choose any available DA to issue requests to, the load will be shared among DAs. Additional DAs also provide robustness in cases where one fails or becomes overloaded.

Scope. The second mechanism for increasing SLP scalability is scope. A scope is a string used to group resources by location, network, or administrative category. SAs and UAs are by default configured with the scope string "default." Figure 3, for example, shows how a UA issuing requests in the legal department of an organization might find services within that scope, but not in accounts payable. (Note that services may be available in multiple scopes.)

UAs can accumulate DA advertisements to form lists of all available scopes. When no DAs are present, UAs can multicast requests for SA advertisements to create lists of scopes supported by the SAs.

DHCP. SLP agents (UAs, SAs, and DAs) can access non-default scopes via static configuration or

DHCP Option for Service Location, SLP Service Scope Option (79).[4] Because it allows an administrator to easily control the set of services available to a particular client, DHCP is actually the preferred method for configuring SLP. For example, all services and clients in a hotel room could be configured to the scope of that one room. A laptop computer used in a particular room would discover only those services in the room itself. Hotel guests could then locate printers in their own rooms, but not in others to which they have no physical access.

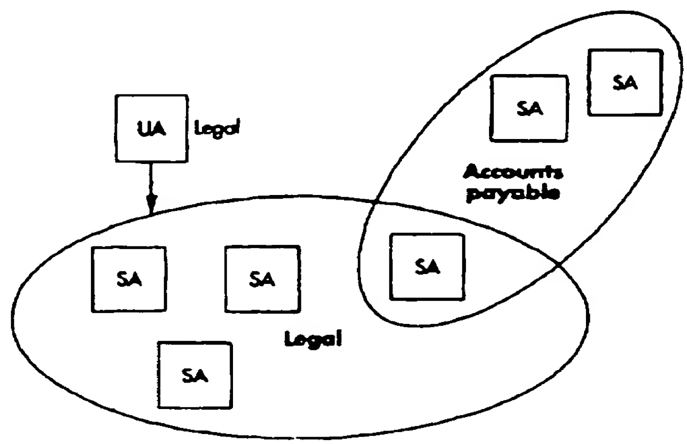


Figure 3. Scope in SLP. User Agents can only locate services within the scopes to which they have access.

Small Office or Home Networks

When there is no directory agent, a UA multicasts the same requests to the SAs that it would have unicast to the DA. Thus, SAs must be prepared to answer Service Requests. A Service Request includes a query that the SA processes against the attributes of the services it advertises. If the multicast request fails to match, or if the SA is unable to process it (due to an error in the request, for instance), the SA simply discards the request.

The multicast convergence algorithm used for DA discovery is also used by UAs for service discovery.

It is important to note that services can be discovered by clients using SLP in small networks without any SLP-specific configuration or the deployment of any additional services. SLP discovery works even in the absence of DNS, DHCP, SLP DAs, and routing. This makes SLP suitable for the home or small office environment, where impromptu and unadministered networks would greatly benefit from automatic service discovery.

FITTING THE PIECES TOGETHER

SLP, in itself, only provides a service discovery framework. That is, SLP agents are idle until service software is advertised, populating the SAs, which in turn propagate information to appropriate DAs. UAs are inactive until a client issues a specific request.

The SLP API, however, allows applications and services to access SLP's functionality. It provides for both synchronous and asynchronous operations, and it features both C and Java bindings. (Table 1 summarizes the C language bindings, and Figure 4 summarizes the Java language bindings.) The API contains separate interfaces for client software to use for discovery and for server software to use for advertising; peer-to-peer software can use both portions. Figure 5 illustrates the interaction between a client and a service, the SLP API, and a UA and SA performing a Service Request operation.

```
public interface Advertiser {
  public abstract void register(ServiceURL url, Vector attributes);
  public abstract void addAttributes(ServiceURL url, Vector attributes);
  public abstract void deleteAttributes (ServiceURL url, Vector attributes);
};

public interface Locator {
  public abstract Locale getLocale();
  public abstract ServiceLocationEnumeration
    findServiceTypes(String namingAuthority, Vector scopes)
  public abstract ServiceLocationEnumeration
    findServices(ServiceType type, Vector scopes, String searchFilter)
  public abstract ServiceLocationEnumeration
    findAttributes(ServiceURL URL, Vector scopes, Vector attributeIDs)
  public abstract ServiceLocationEnumeration
    findAttributes(ServiceType type, Vector scopes, Vector attributeIDs)
};

public class ServiceLocationManager {
  public int getRefreshInterval();
  public static Vector findScopes();
  public static Advertiser getAdvertiser();
  public static Locator getLocator();
};
```

Figure 4. SLP API Java bindings.

Details

SLP is a mostly string-based protocol that uses a binary message header, as shown in Figure 6. Messages are largely composed of UTF-8 strings[8] preceded by length fields. (See Table 2 on page 79 for a description of other fields.)

The SLP header concludes with a Language Tag.[9] Attribute value strings in the SLP message are translated into the language indicated by the tag, and the Service Template[5] associated with a partic-

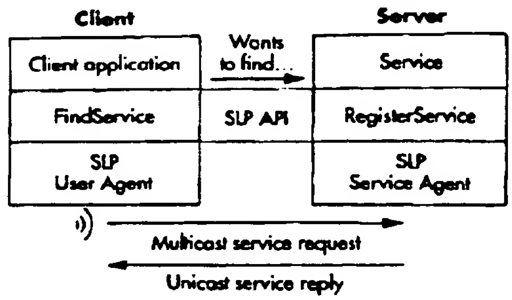


Figure 5. Client server discovery using SLP without DAs. The client application uses a UA to multicast a Service Request that the SA responds to with a unicast Service Reply.

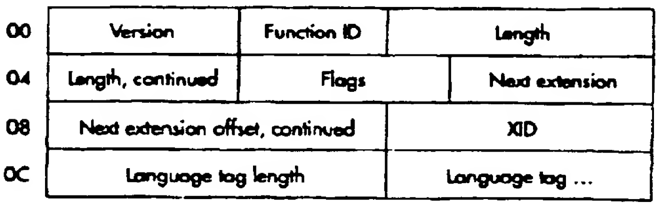


Figure 6. The SLP Header. The SLP header precedes and characterizes all transmitted SLP messages.

ular service provides additional information regarding internationalization. Some strings have standardized translations, while others have fixed meanings and are not intended to be translated.

The Function ID field indicates the SLP message type, all of which are summarized in Table 3 (page 79).

SLP's central function is the exchange of Service Request and Service Reply messages. A UA, for instance, only has to be able to send Service Requests (though it also needs to handle Service Replies and DA advertisements as a result of those requests). An SA need not support any features besides discovering DAs, responding to Service Requests, and sending Service Registration messages to appropriate DAs.

Deployment

Currently, service location information is acquired by prompting the user or reading it from a configuration file. To use SLP, client software vendors must modify the network configuration portions of the client to employ the SLP API[6] to obtain the location of the server. By modifying the failure notification path as well, automatic service discovery can be reinitiated when a server fails. In this way, another server can be located without interrupting the user's service.

Alternatively, it is possible to utilize the benefits of SLP without modifying the client software, as long as the client already uses an existing service for configuration. For instance, some clients use the LDAPv3 protocol[10] to access a directory for configuration information, so they can discover services that SLP has automatically registered with the directory.

SLP attributes, Service Templates, and search filters are all compatible with a subset of LDAPv3. This means that services registered with an SLP DA can be automatically registered into an LDAPv3 directory. That is, an LDAPv3 directory can function as the back end for an SLP DA. Thus, users of the LDAPv3 directory can obtain current network service information automatically. SLP's interoperability with LDAPv3 eases the integration of network configuration information in IP networks, where directories are increasingly used to centralize access.

Table 1. SLP API C language bindings.

Binding	Description
SLPOpen	Clients and services initialize the SLP library and obtain a handle to use with all subsequent calls.
SLPClose	Clients and services release the SLP library.
SLPReg	Services register their service URL and attributes. A service may also use this interface to update its service attributes or refresh a registration before it expires.
SLPDereg	Services can deregister their availability.
SLPDelAttrs	Services can deregister a particular attribute.
SLPFindSrvs	Clients can obtain service URLs based on their query by service type, SLP scope, and/or service attributes. These URLs will, by definition, be the locations of services the client can use.
SLPFindSrvTypes	Clients can discover all types of service available on the network.
SLPFindAttrs	Clients can discover attributes of a particular service or the attributes of all services of a given service type.

There is a great deal of work going on in the area of autoconfiguration. Comparison with other approaches shows SLP's generality and versatility.

DHCP Service Options

The location of several types of service can be configured using DHCP. Administrators can configure certain clients for a particular server. For example, DHCP option 42 configures a host to use a particular Network Time Protocol (NTP)[1] server or servers.

Using DHCP to configure services is significantly different from using SLP. DHCP servers, for instance, have no intrinsic way to determine whether an address actually refers to a currently available server. SLP, on the other hand, lets you discover servers with known availability. SLP also allows a client to discover a server that meets its specific requirements. DHCP provides no such mechanism.

DNS Resource Records for Specifying the Location of Services

The Domain Name System (DNS) SRV Resource Record (SRV RR)[2,3] allows for lookups of domain names associated with service names. Thus, a DNS resolver can request all instances of a particular service type within a given domain.

For example, to find all instances of TCP-based line printer (LPR) services in the domain "nonexistent.net," a DNS resolver would send a request to the DNS SRV RR for the service named "lpr.tcp.nonexistent.net." This might return two domain names: "big.nonexistent.net" and "small.nonexistent.net."

Unlike operations in SLP, resolving a DNS SRV RR currently requires a DNS server to be present. Woodcock and Manning currently have an effort underway to standardize a new mechanism to allow multicasting of DNS requests.[4] According to this proposal, individual systems could contain "stub" DNS servers that would respond to multicast requests in the absence of a true DNS server. This mechanism would remove the requirement for a DNS server to be present, making the DNS SRV RR's approach suitable for small networks lacking in administration and infrastructure.

This service discovery method, however, allows the client to discover services only by type, and not by service characteristics.

There is currently no way to update DNS servers when services become available or go down; as with DHCP, the client system might easily obtain locations for services that are not currently available.

Simple Multicast Discovery Protocols

A variety of simple multicast discovery protocols have been proposed over the years.[5-7] In all of them, a client multicasts a request for a desired service type. All available services that receive the multicast request send a response, including the location of services matching the type requested.

Some of the proposals allow services to announce their presence as they come up and periodically thereafter, so clients can become immediately aware of new services. However, none of them scales beyond a small network. Unlike SLP, they provide no means for automatic service information collection. Moreover, they cannot detect that they are in a larger network and should stop multicasting or limit their TTL to avoid disrupting operations.

Finally, these protocols fail to include any features for reliable multicasting. If dozens of responders attempt to reply to a multicast request, the implosion of replies may inundate the requester. SLP ameliorates this problem by using previous-

continued on p. 78

ADDITIONAL FEATURES

The essential function of SLP is service discovery. SLP has also been designed to provide security, extensibility, support for browsing operations, and operation over IPv6. These features extend the utility of SLP, and will be especially useful once a standardized security infrastructure has been widely adopted on IP networks.

Security

SLP is designed to make service information available, and it contains no mechanisms to restrict access to this information. Its only security property is authentication of the source of information, which prevents SLP from being used to maliciously propagate false information about the location of services.

Digital signatures. An SA can include a digital signature produced with public key cryptography along with its registration messages. A DA can then verify the signature before registering or deregistering any service information on the SA's behalf. These digital signatures are then forwarded in reply messages to UAs, so they can reject unsigned or incorrectly signed service information. Of course, DAs and UAs can only verify signatures, not produce them.

continued from p. 77

responder lists in its multicast convergence algorithm (as discussed in the sidebar). Of course, the replies could be staggered by requiring responders to wait for a random interval, but this would force the requester to wait much longer for answers where there are few results.

Jini

Sun Microsystems' Jini technology provides a Java-oriented set of mechanisms and programmer interfaces for automatic configuration. As an IETF standard, SLP is a general mechanism suited to heterogeneous systems. Jini, on the other hand, leverages Java's uniformity across platforms, providing powerful semantics for service discovery operations.

The Jini discovery architecture is similar to that of SLP. Jini agents discover the existence of a Jini Lookup Server, which collects service advertisements in a manner analogous to DAs in SLP. Jini agents then request services on behalf of client software by communicating with the Lookup Server. Unlike SLP, however, where DAs are optional, Jini requires the presence of one or more Lookup Servers.

Jini's discovery mechanism offers some advantages to Java-based clients. The Lookup Server uses object-oriented matching to determine which services support the client's requested Java interface. Both Jini and SLP use attributes to find services that match the client's requirements, but where SLP uses string-based attributes and weak typing, Jini employs Java objects throughout.

Service discovery with SLP returns a URL denoting a service's location. Jini, on the other hand, returns an object that offers direct access to the service, using an interface known to the client.

For some embedded systems that offer network services, running a Java Virtual Machine may require memory and processing resources that are too costly, thus precluding the use of Jini to advertise services. In those cases, SLP can advertise the services as well as a "Java Driver Factory." A Java Driver Factory is a class that can be used to produce (instantiate and initialize) Java objects based upon initialization parameters. An SLP-Jini bridge can detect services and obtain their attributes and Java Driver Factory. (For more, see http://www.srvloc.org) The bridge uses the Java Driver Factory to instantiate a Java driver object, initializing it with the attributes advertised using SLP. The bridge then registers the service with a Jini Lookup server.

When a Jini client discovers a service, it will be able to use it equally well, regardless of whether it was directly registered with the Jini Lookup Server or registered by proxy via an SLP-Jini bridge. The bridge allows clients to make use of Jini's powerful API to discover services on the network that cannot support Jini natively themselves.

REFERENCES

1. D. Mills, "Network Time Protocol (Version 3). Specification, Implementation and Analysis," RFC 1305, Mar. 1992; available at http://www.rfc-editor.org/rfc/rfc1305.txt.
2. A. Gulbrandsen and P. Vixie, "A DNS RR for Specifying the Location of Services (DNS SRV)," RFC 2052, Oct. 1996; available at http://www.rfc-editor.org/rfc/rfc2052.txt.
3. P. Mockapetris, "Domain Names—Implementation and Specification," RFC 1035, Nov. 1987; available at http://www.rfc-editor.org/rfc/rfc1035.txt.
4. B. Woodcock and B. Manning, "Multicast Discovery of DNS Services," Dec. 1998, work in progress.
5. D. Brown, "Lookup Service," *Personal Communication*, 19 June 1997.
6. S. Hanton, "Simple Server Discovery Protocol," Jan. 1997, work in progress.
7. T. Cai, et al., "Simple Service Discovery Protocol/1.0," Apr. 1999, work in progress.

As an additional level of authentication, DAs can also include digital signatures with their advertisements. UAs and SAs can thus avoid DAs that have not been legitimately established by the site's administration because SLP agents that possess private keys for generating verifiable digital signatures are (by definition) trusted to legitimately advertise themselves.

Security Configuration Requirements. SLP is designed to automatically configure service locations with minimal static configuration requirements for SLP agents. SLP security, however, does require some additional configuration (for the cryptographic keys or certificates used in generating and verifying digital signatures).

When needed, vendors of SLP-enabled clients and services can establish new cryptographic algorithms and data formats within SLP's existing protocol. It is also possible to deploy new keys gradually, without requiring flag days, which would require simultaneous reconfiguration of all interoperating systems. Suppose, for example, that corporate policy requires that old private keys for authenticating servers be replaced in all SAs in the enterprise every three months. If flag days were

required, all SAs, UAs, and DAs would have to be rekeyed at once. SLP, on the other hand, allows phasing in of new keys. SAs include digital signatures generated by both the new and old keys with their messages until all UAs and DAs replace the old keys, at which point, they phase out the old generation of keys.

Extensibility

SLP extensions are additional protocol elements appended to messages. For example, there is an extension for reporting when a service request omits an attribute that is defined as required by a Service Template.[5] Another extension currently under development would allow UAs to request notification of additional services as they appear on the network.

When an SLP agent recognizes a message extension, it will perform the appropriate processing. If the extension is not recognized, it is either ignored or the entire SLP message is discarded, depending on how the message extension is labeled.

SLP's extensibility allows for future enhancements—such as additional error reporting, added notification facility, and so on—without altering the base protocol.

Browsing Features

In addition to its required features, the SLP specification describes several optional features that could be used to support sophisticated service browsers.

Service Type Request. Using this type of message, UAs can discover all service types available on the network. The response supplies a top-level taxonomy of services, which supports the basic requirements for building a general "service browser" on top of SLP.

Attribute Request. A UA can use the Attribute Request to retrieve all the attributes of a given service in a manner similar to a directory lookup operation. The UA can also issue the request *without* naming a specific service instance. The response from the DA (or SAs if there is no DA) returns all attributes and values for the requested service type within the network. For example, an Attribute Request for all available video servers on the network might return the following properties:

- locations include my building and a remote office;
- video streams include programs A and B.

Table 2. SLP header fields.

Header Field	Description
Version	SLP protocol version: 1 and 2 are defined.
Length	Length of the entire SLP message.
Function ID	Message type that follows the SLP Header.
Flags	Indicate special treatment of the message.
Next Extension Offset	Offset, in bytes, to the first SLP Extension.
XID	Unique number for each unique request.
Language Tag Length	Length of the Language Tag that follows.
Language Tag	Indicates the language of all human-readable strings included in the SLP message.

Table 3. SLP message types and descriptions.

SLP Message Type	ID	Description
Service Request	1	UAs find service by type, scope, and search filter.
Service Reply	2	DA (or SA) returns Service URLs and their lifetimes.
Service Register	3	SAs register Service URLs and attributes.
Service Deregister	4	SAs deregister Service URLs and attributes.
Service Acknowledgment	5	DAs acknowledge a successful registration or deregistration.
Attribute Request	6	UAs find attributes by service type or by Service URL.
Attribute Reply	7	DA (or SA) returns attribute information.
DAAdvert	8	DA sends its Service URL, scope, and attributes.
Service Type Request	9	UAs find service types by scope.
Service Type Reply	10	DA (or SA) returns a list of service types.
SAAdvert	11	SA sends its Service URL, scope, and attributes.

With this information, a browser interface could help me determine the location of a video server in my building or a video server that serves video stream A. If, however, I request a server that is in my building and serves video stream A, I might not succeed because the attributes are independent: video stream A might only be available from the server in the remote office.

Attribute Request messages are also useful for determining the attributes of a particular service. The UA locates the corresponding Service URL,

and the Attribute Request uses it to look up the service's attributes. Network software could also discover a particular service's features by requesting the attributes directly, which would require subsequent protocol feature negotiation between client and server.

SLP Operation over IPv6

The formal specification has not yet been standardized, but SLP is designed to provide service discovery facilities that will work for networks using IPv6.[11] Once the debate is settled regarding which string representation to use in URLs for numerical IPv6 addresses, some minor changes will be needed. Service URLs[5] containing numerical addresses will require a different format from what IPv4 uses, and link-local addresses will require some special handling in IPv6. For example, DAs that obtain service registrations with link-local numerical addresses must not forward them using the link on which they were registered. Also, the address to use for site-local scoped multicast operations differs in IPv4 from what it is in IPv6.[12]

SUMMARY

SLP is an IETF standard for service discovery and automatic configuration of clients. It provides for fully decentralized operation and scales from a small, unadministered network to an enterprise network where policy may dictate who should discover which resources. This paper describes how SLP operates and how it adapts to conditions where infrastructure is not available, where administration is minimized, and where network administrators in large enterprises wish to reduce tedium and workload. While alternative mechanisms exist, SLP remains the most general and versatile solution for service discovery on TCP/IP networks. ■

REFERENCES

1. J. Veizades, E. Guttman, and C. Perkins, "Service Location Protocol," IETF, RFC 2165, June 1997; available at http://www.rfc-editor.org/rfc/rfc2165.txt.
2. E. Guttman, C. Perkins, J. Veizades, and M. Day, "Service Location Protocol, Version 2," IETF, RFC 2608, June 1999; available at http://www.rfc-editor.org/rfc/rfc2608.txt.
3. R. Droms, "Dynamic Host Configuration Protocol," IETF, RFC 2131, Mar. 1997; available at http://www.rfc-editor.org/rfc/rfc2131.txt.
4. C. Perkins and E. Guttman, "DHCP Options for Service Location Protocol," IETF, RFC 2610, June 1999; available at http://www.rfc-editor.org/rfc/rfc2610.txt.
5. E. Guttman, C. Perkins, and J. Kempf, "Service Templates and Service: Schemes," IETF, RFC 2609, June 1999; available at http://www.rfc-editor.org/rfc/rfc2609.txt.
6. J. Kempf and E. Guttman, "An API for Service Location," IETF, RFC 2614, June 1999; available at http://www.rfc-editor.org/rfc/rfc2614.txt.
7. T. Howes, "The String Representation of LDAP Search Filters," IETF, RFC 2254, Dec. 1997; available at http://www.rfc-editor.org/rfc/rfc2254.txt.
8. F. Yergeau, "UTF-8, a Transformation Format of ISO 10646," IETF, RFC 2279, Jan. 1998; available at http://www.rfc-editor.org/rfc/rfc2279.txt.
9. H. Alvestrand, "Tags for the Identification of Languages," IETF, RFC 1766, Mar. 1995; available at http://www.rfc-editor.org/rfc/rfc1766.txt.
10. M. Wahl, T. Howes, and S. Kille, "Lightweight Directory Access Protocol, version 3," IETF, RFC 2251, Dec. 1997; available at http://www.rfc-editor.org/rfc/rfc2251.txt.
11. E. Guttman and J. Veizades, "Service Location Protocol Modifications for IPv6," Oct 1998, work in progress.
12. R. Hinden and S. Deering, "IP Version 6 Multicast Address Assignments," IETF, RFC 2375, July 1998; available at http://www.rfc-editor.org/rfc/rfc2375.txt.

FURTHER READING ON SLP

J. Kempf and P. St.Pierre, *Service Location Protocol for Enterprise Networks*, John Wiley & Sons, 1999.

Service Location Protocol Home Page • http://www.srvloc.org/.

Erik Guttman is a staff engineer at Sun Microsystems. He is a member of the Advanced Network Development team in Sun Labs. His technical interests include automatic configuration, network security, and network software testing. He is an active member of the Internet Engineering Task Force where he is the chairman of the Service Location Protocol (SVRLOC) Working Group. He received a BA in Philosophy and Computer Science from UC Berkeley and an MS in Computer Science from Stanford University.

Readers can contact Erik Guttman at erik.guttman@sun.com.